JUL 1991

BEGINNERS' WORKSHOP

Boost your MS-DOS environment in this month's episode of the back to basics series. Mike Hardaker is your guide to 'environment variables'.

MS-DOS has certain default elements which are collectively known as its 'environment'. One of these elements investigated in an earlier episode of *Beginners' Workshop* is the prompt.

The MS-DOS prompt can be changed by using the appropriately-named PROMPT command. It can be altered from its default 'C>' (assuming the C-drive is your current drive) to something a little more informative, like 'C:\DOS>'. This effect was produced by typing **PROMPT PG** to tell us the current directory.

However, PROMPT is not the only MS-DOS command which can be used to alter the environment defaults. Another example is SET.

SET

The SET command is used to store an 'environment variable' ('variable' because it can be altered) in the environment. The prompt is an example of an environment variable, but because it is changed so often, MS-DOS provides a special command for it (PROMPT). You can use SET to define the path variable if you wish to, however. Try typing in the following:

```
SET PROMPT=$P$G
```

This simple example demonstrates the syntax of the SET command quite clearly – you type **SET** followed by the variable that you wish to define (in this case the prompt), with an equals sign [=] and then the value that you want the variable to hold (PG).

You can use SET to establish the value of other environment variables which are automatically supplied by MS-DOS. One of these is the directory path (although this is easier to set using the PATH command as described in an earlier *Beginners' Workshop*), and another is the 'comspec'.

Comspec tells MS-DOS where to look when it needs to find the 'command processor', which is the MS-DOS file that actually takes your instructions and translates them into terms understood by the lower levels of the operating system.

By default, applications will look for this (usually called COMMAND.COM) in the root directory of the drive your system is booted from. However, using SET you can tell the system to look elsewhere if you prefer.

The command below will tell MS-DOS that the command processor has the default name (COMMAND.COM) and that it is in the D:\DOS directory:

```
SET COMSPEC=D:\DOS\COMMAND.COM
```

If you wish to, you can also change the name – essential if you are using one of the 'alternative command processors' such as the popular *4DOS* program.

In order to actually get started, MS-DOS will need a second copy of COMMAND.COM in the root directory of the boot drive. However, if you create a specialised comspec with SET, the earliest place you can put it is in the AUTOEXEC.BAT file; a file which is interpreted by the command processor and must therefore be loaded first. Is this a catch 22 situation?

Not necessarily. There is another way of defining the comspec – by using the CONFIG.SYS file (which is handled before the command processor is loaded). While this method is slightly confusing, as it makes no mention of the word 'comspec', it does work.

You can create virtually the same effect as when you typed:

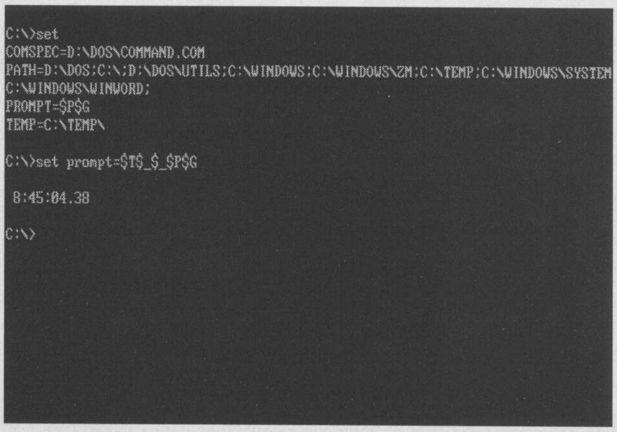
```
SET COMSPEC=D:\DOS\COMMAND.COM
```

by typing the following in your CONFIG.SYS file:

```
SHELL=D:\DOS\COMMAND.COM
```

The only difference is that you don't need a duplicate command processor ('shell' is an alternative name for the command processor, which is also known as the command interpreter).

However, if you can set the environment variables with PROMPT, SHELL and PATH, what's the point of SET? Well, although those are the major environment variables defined by the system, there can actually be a more or less infinite number of variables in the environment. This is simply because you



```
C:\>set
COMSPEC=D:\DOS\COMMAND.COM
PATH=D:\DOS;C:\;D:\DOS\UTILS;C:\WINDOWS;C:\WINDOWS\ZM;C:\TEMP;C:\WINDOWS\SYSTEM
C:\WINDOWS\WINWORD;
PROMPT=$P$G
TEMP=C:\TEMP\

C:\>set prompt=$T$_$_$P$G

 8:45:04.38

C:\>
```

● Using SET, you can tell your computer to look anywhere in the system for certain commands. In this example, the system is looking in the D:\DOS directory for Comspec, which tells MS-DOS where it can find the command interpreter. SET also allows you to customise other elements such as the prompt command, making your system fit your needs.

USING PARAMETERS IN BATCH FILES

It's not only the system which can have variables in MS-DOS, you can also include variables (or 'parameters') which are specific to batch files. Since many MS-DOS commands (and application programs) allow you to add parameters to the command, this concept should be pretty familiar. For example, when you use the TYPE command, you add the filename of the file you want to read as the parameter. So `TYPE LETTER.DOC` has `LETTER.DOC` as a parameter.

TYPE is a fairly irritating command, as it doesn't pause after every screenful of information. Using parameters, we can create a batch file which will introduce that pause, yet will accept a filename as a parameter – an improved version of TYPE, in fact. Create a batch file which looks like this, which is called T.BAT:

```
ECHO OFF
TYPE %1 |MORE
```

`%1` is a special term understood by MS-DOS, which stands for 'the

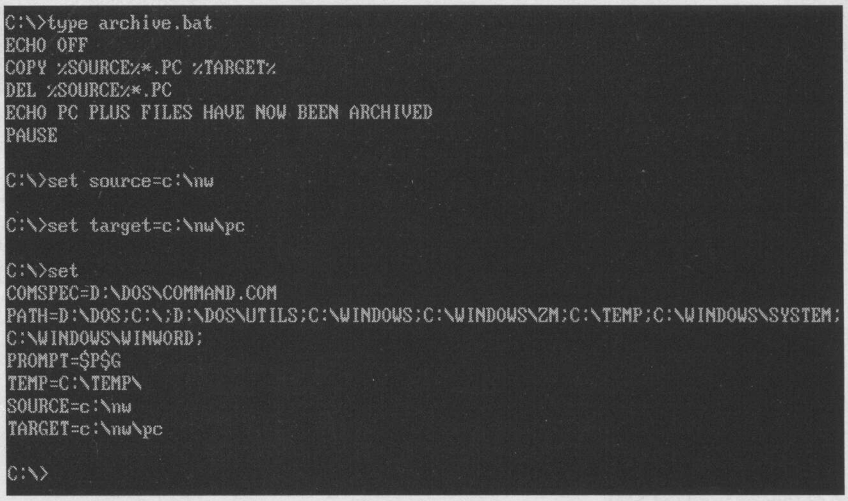
➧

JUL 1991

can define your own variables.

In fact, environment variables are used in this way in many software packages, such as Microsoft *Windows*. Since many *Windows* applications need to know exactly where the main *Windows* files are held, *Windows 3.0* defines an environment variable called WINDIR, which gives the location of the main directory.

Any *Windows* application can then look up the value of that variable and afterwards find the files in that directory. Similarly, *WordPerfect* uses a variable called WP, which contains the parameters that you want to be applied to the program when you start it up.



```
C:\>type archive.bat
ECHO OFF
COPY %SOURCE%*.PC %TARGET%
DEL %SOURCE%*.PC
ECHO PC PLUS FILES HAVE NOW BEEN ARCHIVED
PAUSE

C:\>set source=c:\nw

C:\>set target=c:\nw\pc

C:\>set
COMSPEC=D:\DOS\COMMAND.COM
PATH=D:\DOS;C:\;D:\DOS\UTILS;C:\WINDOWS;C:\WINDOWS\ZM;C:\TEMP;C:\WINDOWS\SYSTEM;
C:\WINDOWS\WINWORD;
PROMPT=$P$G
TEMP=C:\TEMP\
SOURCE=c:\nw
TARGET=c:\nw\pc

C:\>
```

● Make moving files easier by changing the first three lines of your batch file, with the help of SET

USING SET

Environment variables can also be created for your own use – particularly in batch files. This is where the **SET** command really comes into its own.

Before creating your own variables, have a good look at the system environment as it currently stands. This is very easy to do – simply enter, without any parameters:

```
SET
```

You will then get a list of the environment variables.

Typically, you will have COMSPEC, PROMPT and PATH variables with the possibility of several more (depending on which applications you normally use). To begin with, let's create a completely meaningless variable called 'myvar'. Enter:

```
SET MYVAR=Jabberwock
```

You can see **MYVAR=Jabberwock** added to the list of environment variables by entering **SET.** Unfortunately, this isn't particularly useful, as it merely lets another program refer to **MYVAR** and actually uses the text string **Jabberwock** – something that few programs are likely to find essential to their operation. However, it does show how to create an environment variable, and you can clear this variable by entering:

```
MYVAR=
```

Note that nothing should be keyed in after the equals sign. To show that **MYVAR** is no longer on the list of variables, enter **SET** once again. So far so good – now let's do something useful with environment variables. You may possibly remember that we created a batch file a few episodes ago to archive data files to their own directories using the COPY command:

```
ECHO OFF
COPY C:\NW\*.PC C:\NW\PC
DEL C:\NW\*.PC
ECHO PC PLUS FILES HAVE BEEN ARCHIVED
PAUSE
```

This moved all the files with the .PC extension from the C:\NW directory to the C:\NW\PC directory. If we change the names of the source and target directories, however, we then need to change the batch file, which is a rather messy business. Alternatively, we can actually change the batch file itself, so that its first three lines look like this:

```
ECHO OFF
COPY %SOURCE%*.PC %TARGET%
DEL %SOURCE%*.PC
```

This changes the directory descriptions to two variables (the MS-DOS batch language knows they're variables because they have percentage signs around them) called 'source' and 'target'. To make the new batch file work in just the way that the old one did, enter the following:

```
SET SOURCE=C:\NW\
SET TARGET=C:\NW\PC
```

However, if we want to change things, we can follow a cleaner process. Let's say that my word processor has changed from *New Word* to *Word for Windows*. I can actually alter my entire archiving routine to keep pace with my word processor upgrade by simply changing the environment variables to:

```
SET SOURCE=C:\WINWORD\
SET TARGET=C:\WINWORD\PC
```

●

first command line parameter', in this case the file name. To get `LETTER.DOC` shown to you one screen at a time, enter:

```
T LETTER.DOC
```

T.BAT only accepts one parameter. You can have more than one parameter. Here's an example of a batch which uses two, MOVE.BAT:

```
]ECHO OFF
COPY %1 %2
DEL %1
```

This copies the first parameter (`%1`, a filename or group of filenames) to the second (`%2`, a filename or path) and then deletes the source file. You use it in the same way as you use the MS-DOS COPY command, although it performs a slightly different function.

Furthermore, you can have command line variables up to `%9`, if you want, giving eight in all (`%0` is reserved, and stands for the batch file actually being run, so a batch file called TEST.BAT containing ECHO THIS IS %0 would show THIS IS TEST.BAT on screen – %0 is also useful if you want a batch file to 'loop' by calling itself).

Finally, create a batch file which allows you to show the directory listing as you like it, using `%1` for the directory that will be listed and adding any filters you want. For example, here's my MIKEDIR.BAT:

```
ECHO OFF
DIR %1 ORT |MORE
```

HELP US TO HELP YOU

Why not take this opportunity to ask all those niggling questions to which you have been unable to find a satisfactory answer.

There's space here for three but if your list is longer, please write them on a separate sheet. Answers will either be incorporated into future articles or answered on this page.

1 ....

....

2 ....

....

3 ....

....

Please clip out this form (or use a photocopy) and send it to:

Beginners' Workshop,
PC PLUS,
Beauford Court,
30 Monmouth Street,
Bath
BA1 2BW.

Please print clearly:

Name..

Daytime Tel..